Interval of Validity Service
IOVSvc

ATLAS Software Week
Architecture Session

11/18/2002

Charles Leggett <CGLeggett@lbl.gov>

Interval of Validity Service

- Purpose:
  - associates valid time ranges with objects
  - triggers updates of data, and validity ranges when object enters a new validity range
  - allows use of call back functions
- Use cases:
  - alignment data
  - calibration objects
  - detector description information
  - anything which has a timed validity range associated with it

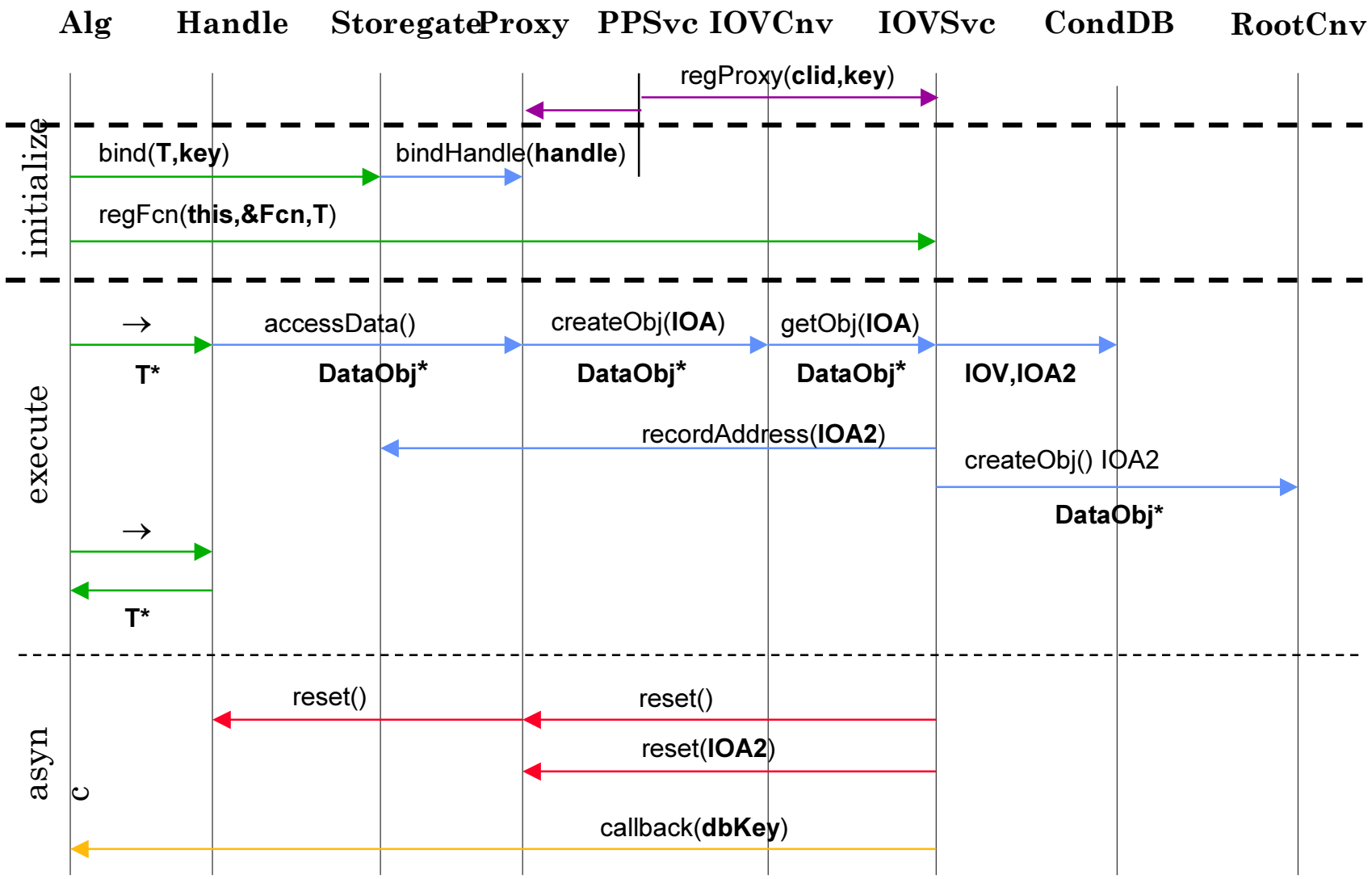
Access Patterns

- Two types of access/usage patterns:
  - pure data, held in dB, used as a data member inside a class. Only needs to have it's contents refreshed when it enters a new validity range.
  - Container class, eg GeoNode, which has no representation inside the dB, contains one or more IOV data objects from dB as data members. Needs to know when any of its constituent members enters a new validity range, at which point a callback method is triggered.
- No overlap between these patterns
  - will never have an IOV data object which needs a callback
  - will never have a container class which has data stored in dB

Class Example


```
class myIOVClass {
   public:
      StatusCode CallBackFcn(list<string> &dbKeys);

   private:
      DataHandle<CalAlign> m_align1;
      DataHandle<CalAlign> m_align2;


      myIOVAlg2* that;
};


myIOVClass::initialize() {

   p_SGSvc->bind(m_align1,dBkey1);
   p_SGSvc->bind(m_align2,dBkey2);

   p_IOVSvc->regFcn(this, &myIOVClass::CallBackFcn, m_align1);
   p_IOVSvc->regFcn(this, &myIOVClass::CallBackFcn, m_align2);


   p_IOVSvc->regFcn(this, &myIOVClass::CallBackFcn,
                    that, &myIOVClass2::OtherCallBackFcn);
}
```


Sequence Diagram



Sequence

- ProxyProviderSvc registers Proxy (dBkey) with IOVSvc
- In `Initialize()`, alg binds `DataHandle<alignData>` with StoreGate, using the same dBkey as the PPS
- Algorithm registers container class (eg GeoNode) with `IOVSvc->regFcn()`, supplying callback fcn and DataHandles as parameters.

- At first deref of the DataHandle, call is passed through to Proxy, which calls `createObj(IOA)` on IOVCnv converter. IOVCnv calls `getObj(IOA)` on IOVSvc, which first talks to CondDB to retrieve storage location of an element given a (Clid,dbKey) pair. It gets back an IOV and an IOA which points to the actual location of the data. The IOVSvc adds the IOV to its list of registered intervals. The IOSvc then calls `createObj()` on the persistency converter for the actual data using this information, and gets back a DataObj*. The DataObj* is passed back through the chain of sequences to the Algorithm, eventually being converted to a T*.

Sequence (cont)

- The next time the DataHandle is derefed, a T* is automatically returned from the Proxy.
- At the start of each event, the IOVSvc parses its list of registered entries to see if any are out of range. When it finds one, it first resets the corresponding proxy, then triggers the registered callback function. The callback function gets a list of the keys of the items which have gone out of range as a parameter.
- The next time one of the DataHandles is derefed, the initial sequence is replayed, getting back a new IOV and data.

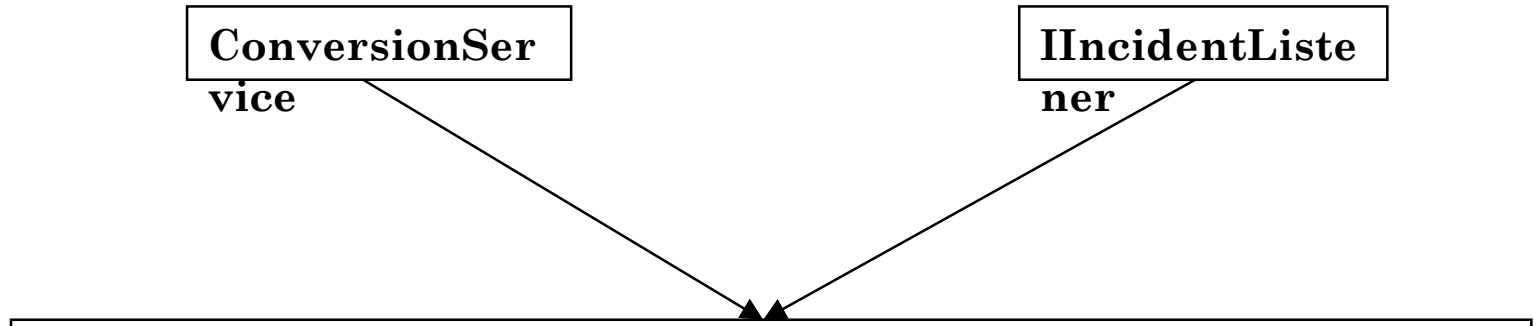
Things Needed

- Proxy Provider Service
- StoreGate::bind(T,key) - slightly different from retrieve()
- Additions to caching policy of DataHandle. Doing a simple reset on the current Proxy isn't enough - the DataHandle keeps its own cache of the data. So the handle has to be informed that its cached data is invalid, or do a check itself every time to see if its Proxy has been reset. Latter is easier, since there can be a many to one relationship between handles and proxies, and the proxies would have to keep track of all the handles pointing to them, but involves an extra dynamic cast (2 of them really) as the Proxy doesn't know the type of the DataHandle pointing to it. Best to have a publish/subscribe relationship between DataHandle and Proxy, so Proxy can reset Handle when needed.

Changes to DataHandle *et al*

- DataHandle
  - inherits from **`IResetable`**, declares reset() method
- DataProxy
  - list of bound handles (`list<IResetable*>`)
  - `bindHandle(IResetable*), unbindHandle(IResetable*)`
  - `resetBoundHandles()`
- StoreGate
  - ProxyProvider
  - `bind()`
    - like retrieve, but also associates Handles with Proxy

IOVSvc




```
IOVSvc {
public:
   StatusCode regProxy(DataProxy *);
   StatusCode regFcn(void (T::*fcn)(), T* obj, DataHandle<H> handle);
   StatusCode regFcn(void (T1::*fcn1)(), T1* obj, void (T2::*fcn2)(), T2* obj);
private:
   set<DataProxy*> m_proxies;
   map<DataProxy*, IOVEntries* > m_entries;
   multimap<DataProxy*, boost::function<StatusCode>* > m_proxyMap;
   multiset<IOVEntry, IOVEntryStartCriterion>  m_startSet;
   multiset<IOVEntry, IOVEntryStopCriterion> m_stopSet;
}
```

Associated Classes

IOVTime

```
IOVTime(int)
IOVTime(EventID)
unsigned long long m_time
```

IOVRange

```
IOVTime m_start
IOVTime m_stop
```

IOVEntry

```
DataProxy* m_proxy
IOVRange*  m_range
```


Internals of IOVSvc

- indexing is done via `DataProxy*`
- IOVs are held in `map<DataProxy*, IOVEntry*>`
- Callback functions are
  `multimap<DataProxy*, boost::function<StatusCode>* >`
- IOVs are ordered in two sets:

```
std::multiset<IOVEntry*,IOVEntryStartCritereon> m_startSet;
std::multiset<IOVEntry*,IOVEntryStopCritereon>  m_stopSet;
```

`m_startSet` is ordered by decreasing start time, `m_stopSet` is ordered by increasing end time. As a result of this ordering, when the sets are scanned at the beginning of each event for entries that need to be marked invalid, as soon as the first valid entry is found in each set, the scanning can be stopped as one is assured that all subsequent entries are valid.

- When invalid entries in these multisets are found, the associated DataProxy* is added to a list. At the end of the scan, the IOVs of these DataProxies are updated, the DataProxies are reset (and thus the associated DataHandles) and the associated callback functions are activated.

What's Next

- This is still a prototype.
- We haven't really decided yet the best way to do the connection between the DataHandle and DataProxy so that the reset is properly done.
- Interface to conditionsDB still needs to be refined.
- Plan to have it in 6.0.0